

CMR Cloud ROM Estimates

All estimates are in perfect man days. Note we are only estimating changes since the original ROM.

Estimates to Remove

- All prototyping estimates
- Metadata-db updates to use S3
- RabbitMQ estimates

New or Updated Estimates

- SNS/SQS
- Database transition
 - DMS
 - ECHO schema transition if prototyping indicates DMS will not work
- Workload drivers
- NGAP integration
- Elasticsearch EC2 instance patching

Test Using AWS Database Migration Services for DB Synchronization (10)

Note: We plan to execute this test prior to the start of Phase 2. This estimate should not be included in the Phase 2 ROM

- Setting up DMS instance (5)
 - Working with Ops/Security to open ports to allow DMS to connect to workload db. (2)
 - Initial DMS instance setup (1)
 - Initial import of data from Workload to RDS (1)
 - Enabling DMS (1)
- Creating Snapshot of data and setting up second RDS instance for testing (1)
- Testing second RDS instance functionality using CMR search/ingest. (2)
- Testing first instance to verify that DMS can keep up with production database. (2)
 - Start a workload ingest run.
 - Measure delay between concepts being ingested into workload database and same concepts appearing in AWS RDS
 - Confirm that delay is not monotonically increasing with time (is DMS keeping up)

SNS/SQS Development (9)

During the prototyping we determined that using Amazon SNS and SQS was a better solution than setting up EC2 instances running RabbitMQ.

- Code changes to use SNS/SQS instead of RabbitMQ (3)
 - Much of this was already done during prototyping
- Automated queue/topic creation. (3)
- Get all integration tests working in AWS with SNS/SQS (1)
- Workload testing to verify performance (2)

Database Transition

Database Migration Service (DMS) (11)

- Setup DMS to migrate data (1)
- Write scripts to verify databases are in sync
 - Dozens of smaller ECHO Business schema tables (1)
 - Orders and all child tables (2)
 - Sequences (1)
 - Metadata-db (2)
 - Tags / Associations
 - Collections
 - Granules
 - Groups
 - Services
- Perform verifications, identify any bugs, and fix bugs (4)

Backup Plan (38)


We are going to prototype the Database Migration Service (DMS) to determine if the functionality works for migrating all of our schemas and determine if performance is adequate for continuous replication from our on-premises database to Oracle RDS. If the prototype indicates DMS cannot be used we need a different plan.

- Design solution for migrating large ECHO business schema tables including orders and system tokens (2)
- Implement changes
 - For estimate purposes assume the following solution as worst case estimate
 - Setup a new message queue (1)
 - Modify any ECHO services to place a message on the queue with enough information to replicate in AWS (4)
 - Write a consumer to process messages and save the new messages (5)
 - Write a script to verify the database are in sync (3)
- Metadata-db transition (use original CMR Event processing plan and estimate) - (23)

NGAP integration (10)

NGAP will be setting up support for new services required for CMR in parallel with our efforts to move to the cloud. We will likely need to work closely with the NGAP team and help with testing out some of the services with CMR components. There may also be times when we are waiting for a service to be made available and will require temporary workarounds on the CMR side that will take extra time to implement. To lower risk we would estimate 10 days worth of CMR development time to verify NGAP functionality, setup CMR services using NGAP, and implementing workarounds (e.g. setting up EC2 instances using RabbitMQ temporarily while waiting on SNS/SQS approval).

Elasticsearch EC2 instance patching (11)

See  [CMR-2642](#) - JIRA project doesn't exist or you don't have permission to view it. for notes on the plan for handling security patches and elasticsearch upgrades once we move to the cloud.

- Design orchestration with ELBs, search application, and indexer application to use the appropriate elasticsearch cluster (1)
- Indexer endpoint to dynamically configure indexer to index to multiple clusters (or back to a single cluster) (3)
- Modify bootstrap to use a message queue to fully reindex elastic
 - Single producer places messages on a queue to index a range of concept-ids (3)
 - Consumers process messages on the queue
 - Retrieve the concepts' metadata in batches (2)
 - Bulk index to elasticsearch (2)

Orders (5)

- Integrating tests

Workload (33)

- Transition Testing (13)
- Performance tests (20)
 - Write the workload driver code (10)
 - As part of the prototyping phase we used a tool called siege to handle most of our load testing. We determined that it was adequate for the workload performance testing we need to perform in AWS. The one feature siege lacks that we would find useful is to specify timestamps for each message to run.
 - Code to generate siege files for search workload from search log files (2)
 - Much of this was already done during prototyping
 - Code changes to generate curl script for JSON queries that were not working with siege (1)
 - Modify ingest to log metadata as part of the ingest request (1)
 - Code to generate siege files for ingest workload from ingest log files (2)
 - Write scripts to automate kicking off a siege run (1)
 - Code to analyze workload performance and generate HDR histograms based on logs (3)
 - Some of this written during prototyping
 - Run performance tests and fix issues (10)

SIT (6)

- Deploy (1)
- Perform transition in SIT (2)
- SIT Verification (2)
- SIT Switchover (1)

UAT (9)

- Deploy (1)
- Perform transition in UAT (2)
- UAT Verification (5)
- UAT Switchover (1)

Ops (16)

- Deploy (1)
- Perform transition in Ops (3)
- Ops Verification (10)
- Ops Switchover (2)